
Software Packaging for Reuse

Prepared by:
NASA Earth Science Data Systems –
Software Reuse Working Group

February 25, 2011

Version 1.0



Earth Science Data Systems Software Reuse Working Group

Editor:

James J. Marshall (INNOVIM / NASA Goddard Space Flight Center)

Contributing Working Group Members:

Bradford Castalia (University of Arizona)

Robert R. Downs * (Columbia University / NASA Socioeconomic Data and Applications Center)

James J. Marshall (INNOVIM / NASA Goddard Space Flight Center)

Chris A. Mattmann * (NASA Jet Propulsion Laboratory / University of Southern California)

David McComas (NASA Goddard Space Flight Center)

Neal Most (INNOVIM / NASA Goddard Space Flight Center)

Mark Sherman (SGT Inc. / NASA Goddard Space Flight Center)

Cindy Sweitzer (United Space Alliance / NASA Johnson Space Center)

* Co-chairs

Working Group Participants:

Nadine Alameh (MobiLaps LLC)

Armando Barreto (University of Arizona)

Stephen Berrick (NASA Headquarters)

Corey Bettenhausen (Science Systems and Applications, Inc. / NASA Goddard Space Flight Center)

Saurabh Channan (University of Maryland)

Kamel Didan (University of Arizona)

Yonsook Enloe (SGT Inc. / NASA Goddard Space Flight Center)

Stefan Falke (Washington University in St. Louis)

AI Fleig (PITA Analytic Sciences / NASA Goddard Space Flight Center)

Michael Folk (National Center for Supercomputing Applications)

Bill Frakes (Virginia Tech)

Ryan Gerard (INNOVIM / NASA Goddard Space Flight Center)

Larry Gilliam (INNOVIM / NASA Goddard Space Flight Center)

Thomas Huang (NASA Jet Propulsion Laboratory)

Mary Hunter (INNOVIM / NASA Goddard Space Flight Center)

Tommy Jasmin (University of Wisconsin-Madison, Space Science and Engineering Center)

Virginia Kalb (NASA Goddard Space Flight Center)

Marilyn Kaminski (National Snow and Ice Data Center – Boulder)

Louis Kouvaris (Science Applications International Corporation / NASA Goddard Space Flight Center)

Kwo-Sen Kuo (University of Maryland Baltimore County / NASA Goddard Space Flight Center)

Scott Lewis (National Snow and Ice Data Center – Boulder)

Michael Leyton (Rutgers University)

Ron McMiller (Dittmar Associates / NASA Johnson Space Center)

Steve Olding (Everware / NASA Goddard Space Flight Center)

Shahin Samadi (INNOVIM / NASA Goddard Space Flight Center)

Bill Teng (Science Systems and Applications, Inc. / NASA Goddard Space Flight Center)

Curt Tilmes (NASA Goddard Space Flight Center)

Bruce Vollmer (NASA Goddard Space Flight Center)

Christine Whalen (INNOVIM / NASA Goddard Space Flight Center)

Jonathan Wilmot (NASA Goddard Space Flight Center)

Bruce Wilson (Oak Ridge National Laboratory)

Robert Wolfe (NASA Goddard Space Flight Center)

Cynthia Wong (NASA Jet Propulsion Laboratory)

Note: All affiliations above were recorded at the time of the member's contribution or participation.

Acknowledgement:

We are grateful to support from NASA's Earth Science Data System Working Groups and to the contributions of members of the Software Reuse Working Group that helped generate this document.

Recommended Citation:

NASA Earth Science Data Systems Software Reuse Working Group (2010). Software Packaging for Reuse, Version 1.0. February 25, 2011. Available: http://www.esdswg.com/softwarereuse/Resources/guidelines/SRWG-Packaging_v1.0.pdf/view

Table of Contents

1. INTRODUCTION	1
OBJECTIVE	1
CONTEXT	1
BACKGROUND	2
JUSTIFICATION	2
2. THE SOFTWARE PACKAGING DOMAIN MODEL	3
3. SURVEY OF EXISTING PACKAGING APPROACHES.....	4
SOFTWARE PACKAGING ATTRIBUTE DEFINITIONS	4
<i>Packaging Technology</i>	5
<i>Metadata</i>	5
<i>Support</i>	5
<i>Ease of Use</i>	5
<i>Pervasiveness</i>	5
<i>Domain of Use</i>	5
<i>Licensing</i>	5
<i>Applicable Standard</i>	5
<i>Compression Method</i>	6
<i>Compliance & Portability</i>	6
<i>Installer</i>	6
4. SOFTWARE PACKAGING CASE STUDIES.....	8
5. SUMMARY	8
6. REFERENCES	8

1. Introduction

Objective

The objective of this document is to identify and describe the attributes of software packaging and to prescribe a domain model and information architecture for the packaging and dissemination of reusable software assets. In support of the objective, this document makes several contributions. First, it reports back on a survey conducted by the NASA Earth Science Data Systems (ESDS) Software Reuse Working Group (WG). The survey evaluates existing compression technologies and software packaging models, and compares the technologies and models along a set of seven dimensions relevant to software reuse. Based on this survey, an initial domain model and information architecture is culled and presented. The model and information architecture will be discussed in the context of a system for managing reusable software assets called the Reuse Enable System (RES). The RES is a software system developed by the Working Group that is a reference implementation of the packaging model described in this document. It is the goal of the Working Group to continue to improve the packaging model and evaluate it in the context of future case studies and use cases. Currently the RES section is a placeholder for further elaboration of the RES use case. The major contribution of this version of the document is the specification of the packaging model.

Context

The ESDS Working Groups (WGs) were established by NASA in 2004 as a follow-on to the Strategic Evolution of Earth Science Enterprise Data Systems (SEEDS) Study, which built on New Data and Information Systems and Services (NewDISS) concepts of distributed, heterogeneous, measurement-based data systems. Together, these studies emphasize the concept of flexible, distributed system elements, which have considerable implementation freedom, the role of the community in shaping data system practices, and a focus on competition, measurement-based (Principal Investigator) systems, and an evolutionary approach to overall data system management. Four WGs were convened to carry out this charter: (1) Metrics Planning and Reporting, (2) Reuse and Reuse Frameworks, (3) Standards and Interfaces, and (4) Technology Infusion. These groups provide community forums for bringing issues to the table in these topic areas, and individually and jointly identify methods, tools, and resources that offer improvements and solutions to recognized issues. Although the WGs do not make policy, they do send recommendations to NASA Headquarters for consideration.

The WG has looked into the current standards for packaging and existing work, which forms the basis of the work described in this document. The context for this work is that of Earth science data systems and other similar data systems assets, and in particular improving the reusability of such assets through an appropriate packaging model and architecture.

Background

Software packaging refers to the “methodology and technology for assembling and encapsulating the components of a software asset” (NASA Software Reuse Working Group, 2010, p. 7). Packaging is an important factor in the reusability of software assets. The WG identified it as one of nine topic areas important for measuring reusability maturity in the development of Reuse Readiness Levels (RRLs) (NASA Earth Science Data Systems Software Reuse Working Group, 2010). Improvements in software packaging offer benefits to those who package the software as well as those who reuse the software. A simple, well-organized software package is easier to prepare and easier to reuse than a complex, disorganized package. Furthermore, improved packaging can reduce the need to obtain support during the adoption process, which reduces the need to provide support after the asset has been distributed.

The way an asset is packaged influences the complexity of the resources that are distributed as part of the asset, including the documentation. If the packaging is simple and logical, it leads to simpler and easier to understand documentation of the package and the availability of an asset that is easier to install and use, both of which increase its reusability. The more reusable an asset is, the more benefits it is likely to provide, so a well-packaged asset has its advantages.

The packaging of a software asset can contribute to its potential reusability. Attention to packaging can ensure that important aspects of a particular version of a software product are included within that version of the product to ensure that it can be reused, extended, modified, or configured without having to go out and gather the versioning information of the asset. In addition, packaging keeps the necessary parts of the software product together as a single unit and informs the potential user of the items that are included within the package. Preparing the package for a particular version of a software product and including an inventory of the items included in the software package enables potential adopters of the software product to determine whether all of the items of the package have been received. Packaging enables each version of a software product to be distinctly identified and differentiated from other versions in terms of what is included in each version. Furthermore, as previously mentioned, in recognizing the importance of packaging to reusability, packaging has been designated as one of the topic areas of the RRLs that are used to assess the potential reusability of software assets being considered for reuse.

Justification

The WG has explored the topic of packaging for reusable software assets in the past; see the “Bottom-Up Reuse Guidelines: Packaging” item on the WG’s portal web site (<http://www.esdswg.com/softwarereuse/Resources/guidelines/guideline-packaging/>). The WG also has recognized the importance of good packaging to the reusability of an asset, as outlined in the Background section above. Therefore, to assist software developers in improving the way in which they package their reusable software assets, to increase their reusability and thus their benefit to the Earth science data systems community, the WG recommends the adoption of the methods and tools described within this Software Packaging for Reuse document. At present, this document targets software developers, primarily, providing them information on ways they can package their software and the types of components to consider including in the package. Secondly, it can assist

consumers looking for reusable software assets by offering them suggestions on what to look for in a package.

2. The Software Packaging Domain Model

In this section, we describe a domain model of relevant concepts and objects for software packaging. The concepts, relationships and objects all directly influence the overall reusability of a particular software asset, and influence the strategy employed for packaging that asset.

The following list of concepts and relationships all influence a software asset's packaging:

- **Software Asset** – The software asset itself, be it an Executable, Script, Documentation, etc., influences the methodology in which it is packaged. As is shown in Table 1, some packaging technologies (e.g., XML-based) will be less efficient, or outright do not support particular assets (e.g., binaries), while scripts and documentation are directly supported.
- **Versions** – Versions of a software asset may influence its packagability. For example, a particular software asset's version may include large dependencies such as Java libraries that make particular packaging technologies and methodologies that favor compression, whereas other version of that same asset may reduce external dependencies and depend more on the core Java framework, making compression less important as a property to maximize when selecting the packaging technology.
- **Build Tools / executables used to generate the asset (Upstream Pedigree)** – Build Tools and other programs used to generate the asset may directly influence packaging, because they (e.g., Maven in the case of Java, Buildout and Setuptools in the case of Python) may facilitate the use of a particular packaging technology (Zip, etc.) during the build lifecycle.
- **Dependencies (other software needed for executing or rebuilding the asset)** – Runtime and configuration dependencies may be required information to include in a software asset's packagable form.
 - **Assets that are children of this asset (Downstream Pedigree)** – Assets may require other Assets (parent-child relationship) in order to properly execute. These software assets and their information may need to be included in the final package.
- **Packaging Methods** – Technologies and existing packaging standards directly influence the method, compression, and tool support for generating and using a software package.
- **License / Licensing Concerns** – Licenses influence the means and mechanisms wherein which software packages and assets can be later reused and built upon in a software system.
- **Distributor / Distribution Mechanisms** – Existing distribution communities and web sites provide means for disseminating software packages and assets.
- **Maintenance Needs** – The requirements for maintaining a software asset (such as evolutionary requirements, new target platforms) may influence the applicability of approaches and technologies for packaging a software asset.
- **Lifecycle (planned releases, upgrades, etc.)** – The overall lifecycle of an asset may influence the selection of packaging technologies and approaches required to build a software package.

These relationships comprise the packaging information model and are captured in Figure 1 and shown graphically for summarization and visualization.

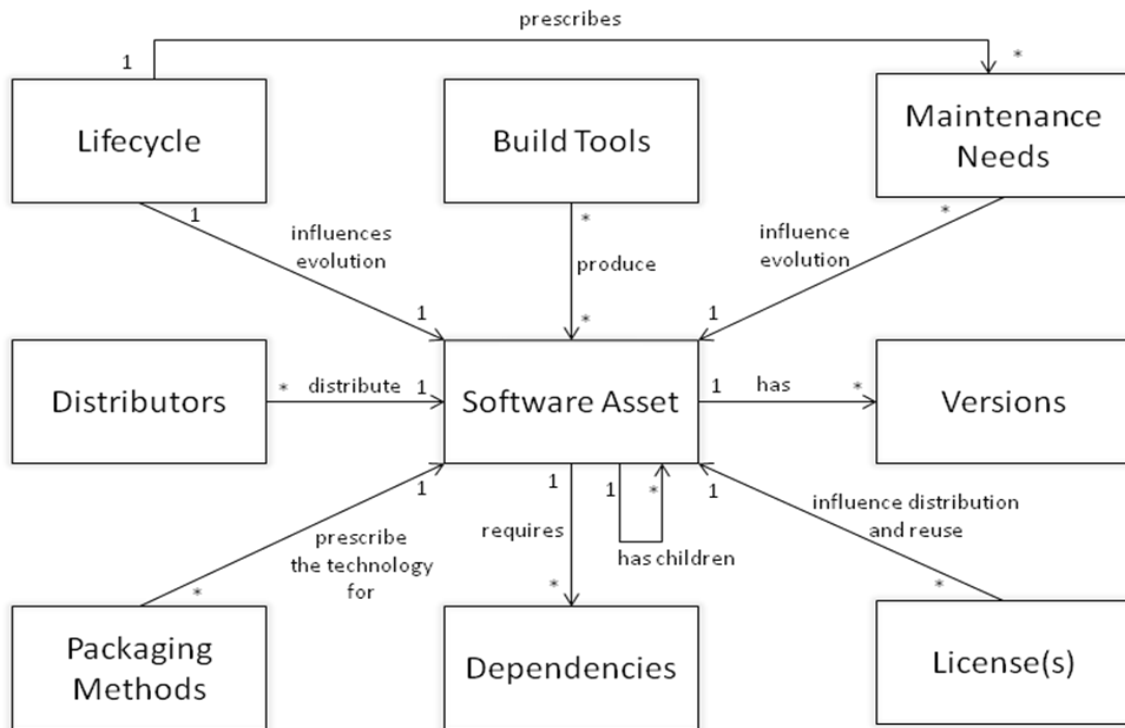


Figure 1 – A software packaging domain information model. Boxes represent concepts (attributes) relevant to the software package. Lines indicate a particular relationship (with labels on the lines spelling out the relationship type). Cardinality of the relationship is indicated by a “1” (meaning a 1-to-1) relation, or by a “*” indicating a 1-to-many relation.

3. Survey of Existing Packaging Approaches

To evaluate our domain model for software packaging, the Working Group decided to perform a survey of existing technological approaches for software packaging. Various technological approaches have been used to package software for reuse, such as zip, jar or ant. Rather than conduct an exhaustive survey of all technological approaches that have been developed for packaging software, selected technological approaches to software packaging are presented in terms of the software packaging attributes, such as Packaging Technology, Maturity and Metadata. This approach offers comparison and may assist with adoption decisions based on an informed understanding of the features and functionality offered by each technological approach. The attributes of software packaging are initially described to provide context and understanding about each attribute, independent of the approach that has been employed by utilization of any particular technology.

Software Packaging Attribute Definitions

We have defined several packaging attributes used to compare software packaging technologies, derived from the information model presented in Figure 1, and its packaging standards attribute/concept. There are several attributes that contribute to technology and

standards selection for the packaging of software assets. The attributes of software packaging are identified and defined below.

Packaging Technology

Packaging Technology refers to the method used to encapsulate the contents of the software and associated information. The technology used to package software can be described by attributes such as metadata, support, ease of use, pervasiveness and maturity, domain of use, domain preference, open source, license, developed by CCSDS, and compression method.

Metadata

The metadata attribute describes the capabilities of the packaging technology to store information about the package. Metadata also includes a file manifest and salient information about the source files, data, configuration files, and third party product bundles that are ingredients forming the package, as well as any packaging scripts that facilitate distribution and deployment of package. It also defines any outside dependencies to enable proper asset operation.

Support

The support attribute describes the level of assistance that is available for users of the packaging technology.

Ease of Use

The ease of use attribute describes the amount of expertise that is necessary to use the package and the packaging technology.

Pervasiveness

The pervasiveness attribute describes the extent to which the technology has been adopted by software development communities.

Domain of Use

The domain of use attribute describes the areas of practice in which the packaging technology has been adopted and where it has been pervasive.

Licensing

Licensing refers to the intellectual property rights and restrictions associated with the packaging technology. Generally, packaging technology that is considered open source is free of legal restrictions on its use.

Applicable Standard

The applicable standard attribute describes whether the packaging technology has been developed under a particular standard or convention.

Compression Method

The compression method attribute describes the software and algorithm used to reduce the volume of the package when the package is created using the packaging technology, if such compression is used at all.

Compliance & Portability

Information that explains any relevant standard(s) the asset complies with or attempts to, as well as the platforms the asset supports or has been shown to port to with the methods used to demonstrate portability.

Installer

The installer refers to the capabilities offered by the software package for installing the packaged software. The installer is sometimes referred to as a setup program. The inclusion of an installer contributes to the ease of use of the packaged software by adopters.

Table 1. Survey of different Packaging Methods

Packaging Technology	Metadata Support	Ease of Use	Pervasiveness	Domain of Use	Open Source	Applicable Standard	Compression Method
<i>Red hat Package Manager (RPM)</i>	Difficult to add metadata other than the existing package metadata defined by RPM.	Command line utility, extensible APIs, C interface.	Used throughout RedHat Linux systems, and available on many UNIX systems as well.	Software Package Installation.	Yes		Zlib
<i>Grid Packaging Tool (GPT)</i>	Any Metadata can be added with little effort. Metadata can be per file, or per package, or both.	Perl scripts and modules, extensible interfaces, command line tool, XML schema to describe software and tools..	Used throughout Grid community to install Grid software.	Grid Software Package Installation.	Yes		Zlib, XML
<i>Xml Formatted Data Units (XFDU)</i>	Any Metadata can be added with little effort. Metadata can be per file or per package, or both.	Java API, command line tool, XML Schemas to describe software and data model.	Emerging Standard, although recently used in DSMS prototype and at ESA.	Generic, meant to be used in any domain.	Yes??	CCSDS	XML, ZIP
<i>Space Formatted Data Units (SFDU)</i>	Supports user-defined metadata attributes, and a base set of metadata.	???	Used by NSSDC as a packaging method for creating AIPs	Space Data Systems.	Yes??	CCSDS	????
<i>zip</i>	Basic metadata such as file size, file modification date, etc. No support for adding metadata by default.	Java, C, C++, APIs, command line interface, extensible.	The de facto standard for compressing files.	Generic packaging technology.	Yes		ZIP
<i>jar</i>	Basic zip metadata, along with META-INF directory describing package contents. META-INF contents may conform to an Apple standard.	Command line interface, Java API interface.	The de facto standard for aggregating and compression binary java classes and packages.	Software Packaging.	Yes		ZIP
<i>tar</i>	File Header Block and other metadata files. Contents follow conventions established by typical use.	Command line interface, API interfaces.	Long legacy in Unix, Linux, Windows; GNU project	File / Software Packaging	Yes (GNU)		ZIP
<i>Mkpkg</i>	Unix	Command line interface	HP's Unix	File / Software Packaging	Yes		?
<i>Apple Disk Image (.dmg)</i>	https://secure.wikimedia.org/wikipedia/en/wiki/Apple_Disk_Image	Utility applications, command line interface	Primarily Apple's Mac OS X, but can be opened in Windows, Linux, Unix	Packaging software for distribution over the Internet	No?		ADC, zlib, bzip2 ?
<i>Windows Installer (MSI)</i>	https://secure.wikimedia.org/wikipedia/en/wiki/Windows_Installer	Commercial and freeware products	Microsoft Windows	Packaging software for installation, maintenance, and removal	No?		None?
<i>Nullsoft Scriptable Install System (NSIS)</i>	https://secure.wikimedia.org/wikipedia/en/wiki/Nullsoft_Scriptable_Install_System	Script-based applications ?	Microsoft Windows	Software package installation	Yes		Zlib, bzip2, LZMA

4. Software Packaging Case Studies

At present, this is a placeholder for future versions of this document wherein which we elaborate on case studies that demonstrate the packaging model. In this version of the document, the major contribution is the elaboration of the packaging model.

5. Summary

The packaging of software for reuse contributes to the reusability of software and has been described to improve understanding of software packaging and software packaging practices. The attributes of software packaging have been identified and described, and an initial domain model and information architecture for software packaging have been introduced.

The attributes of software packaging have been identified and described, independent of any particular technology or implementation, to improve understanding of the attributes and how they can be utilized to improve the reuse of software assets. Such understanding also could foster the creation and adoption of new technologies to improve software packaging practices and contribute to the reusability of software products.

An initial domain model and information architecture for the packaging of reusable software assets was presented. The model was devised by evaluating the results of working in the NASA Earth Science Data Systems (ESDS) Software Reuse Working Group (WG), including the results of a survey into existing packaging and compression technologies, and attribute models for software reuse. The domain model and information architecture for packaging will be evaluated in the context of a specific use case, the NASA Reuse Enablement System (RES), developed by the Working Group to track and manage existing reusable Earth science data system relevant software assets. The RES is a reference implementation of the model and architecture described in this paper. We expect more case studies to be added and the document to be evolved and improved going forward.

6. References

1. NASA Earth Science Data Systems Software Reuse Working Group (2010). Reuse Readiness Levels (RRLs), Version 1.0. April 30, 2010. Available: <http://www.esdswg.org/softwarereuse/Resources/rpls/>
2. <http://www.gnu.org/software/tar/>
3. http://en.wikipedia.org/wiki/Tar_%28file_format%29
4. http://www.hpl.hp.com/personal/Carl_Staelin/mkpkg/